AnyTraffic routing algorithm for label-based forwarding

P. Pedroso*, O. Pedrola*, D. Papadimitriou**, D. Careglio*, M. Klinkowski*
*Technical University of Catalonia, Barcelona, Spain, **Alcatel-Lucent Bell NV, Antwerpen, Belgium
{ppedroso, opedrola, careglio, mklinkow}@upc.edu, Dimitri.Papadimitriou@alcatel-lucent.be

***Abstract*—The high capacity provided by packet-switched networks is supporting the proliferation of bandwidth intensive multimedia applications which require multicasting capability. As a consequence a mixed traffic scenario where both unicast and multicast demands compete for the same shared resources, is the one more likely to be found within the current transport networks. On today's traffic-engineered networks such traffic travels through logical data paths setting up by constrained-based routing schemes provided by a control plane. In this paper we have devised a novel constraint-based routing scheme to forward unicast and multicast traffic envisioning a system resource consumption outstanding performance. We introduce the concept of AnyTraffic data group which consists of a group of egress nodes receiving unicast and multicast traffic over the same single minimum-cost tree. A novel Steiner tree-based heuristic algorithm is specifically defined to accommodate such data group and has been compared with the standard shortest path (SP) algorithm - the optimal case for unicast routing - and a classical Steiner tree (ST) heuristic algorithm - the optimal case for multicast routing. Exhaustive experiments have been done to validate the results.**

I. Introduction

Within the next-generation packet-switched networks bandwidth-intensive multimedia applications such as HDTV, interactive teleconferencing, distributed data processing and video broadcasting are catching on. These new services and applications are requiring multicasting of information from a source to a set of destination nodes in addition to the conventional unicasting of information from a source to a single destination node. Therefore a mixed traffic scenario – where both connection types may demand high bandwidth capacity – is the one more likely to be found in today's meshed aggregation environment. In order to cope with these new service requirements as well as reducing operational costs (CAPEX), Network Providers aim to evolve their transport networks to a more efficient, scalable, and secure packet-switching infrastructure. Traffic-engineering architectures have been deployed in the last past years reflecting such intention namely either multi-protocol label switching (MPLS) [1] architecture or its generalization, (G)MPLS [2]. Making use of the label switching concept these architectures enable a set of advanced traffic engineering (TE) capabilities to optimize the utilization of network resources (resource-oriented performance) and to enhance the QoS of traffic flows (traffic-oriented performance). From the control plane perspective a label switching architecture allows the set up of point-to-point (P2P) and point-to-multipoint (P2MP) data paths using constraint-based routing schemes. At the forwarding plane level it allows fast forwarding as well as distinguish between unicast (one-to-one) and multicast (one-to-many) traffic using simpler header information. In this context, these paths are instantiated as switched data paths.

In this paper we focus on those constraint-based routing schemes able to compute data paths allowing to forward an offered load consisting of combined unicast and multicast traffic in packet-switched meshed networks. Three routing approaches are considered for simulation and analysis. The first and second routing approaches are broadly applied in current switched technologies, namely (1) the set up of a set of P2P switched data paths to encapsulate whether unicast or multicast traffic (i.e., multicast is treated as point-to-point traffic) and (2) the set up of dedicated P2P switched data paths for unicast traffic forwarding and dedicated P2MP switched data paths for multicast traffic forwarding. The third routing approach is an outcome of this study. In this innovative approach, the traffic-based switched data paths of the former approaches are aggregated on a overlap basis and optimized in order to forward unicast and multicast traffic together (i.e., over the same path) as much as possible. For this purpose, we introduce the concept of AnyTraffic data group which consists of a group of egress edge nodes receiving unicast and multicast traffic over the same single minimum-cost network entity (e.g. tree). Note that this novel routing scheme is seamlessly applicable over any existing infrastructure, such as IP/MPLS, Ethernet or any packet-based switching technology [3][4] where the following conditions are met i) at control plane level: root-initiated point-to-multipoint and source-initiated point-to-point switched data path (e.g. RSVP-TE or alike) and ii) at the forwarding plane level: to be capable to distinguish multicast from unicast traffic by inspecting other header information than the destination address (e.g. label/tag).

The objective of this paper is to introduce a novel routing scheme adopting the new concept of AnyTraffic data group and compare it against the other two traditional schemes using extensive simulation experiments. Relevant system metrics as bandwidth and system resource consumption are evaluated. The aim of the novel routing strategy is to benefit from the advantages of the traditional approaches while minimizing their respective drawbacks and to achieve better system resource consumption (for state maintenance). Indeed, from the control plane perspective, each switched data path is identified by a state entrance recorded at the forwarding table of each node

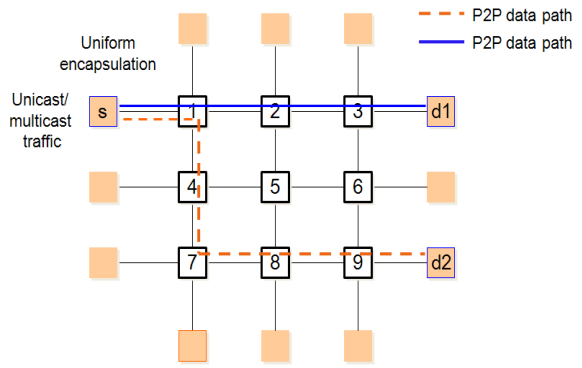

Fig. 1. Approach 1: both unicast and multicast traffic is carried over point-to-point data paths between each edge-node pair. In the example, $(s, d1)$ P2P data path and $(s, d2)$ P2P data path.

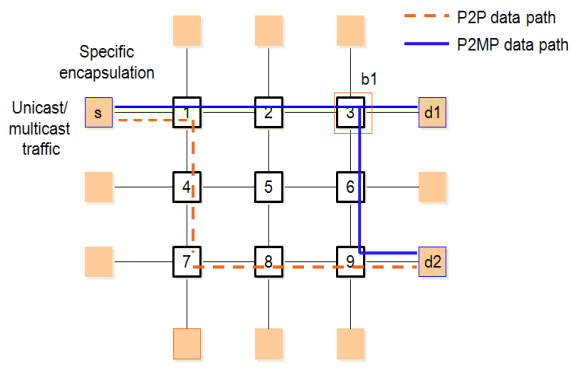

Fig. 2. Approach 2: multicast traffic is mapped to dedicated P2MP data paths and unicast traffic is mapped into dedicated P2P data paths. In the example, (s, g) P2MP data path, $g = \{d1, d2\}$, and $(s, d2)$ P2P data path.

along with the route of the path. Hence we are pursuing the reduction of the total number of states needed to setup and maintain a logical data path by forwarding the traffic together (i.e., over the same path which implies the record of just one state) as much as possible. The impact of such an objective is an increase of both bandwidth consumption and length of unicast path. In order to guarantee a low increase of the former metrics, a threshold have to be found to decide where to separate the unicast route from the multicast route (i.e., the placement of a branch node). In this study a novel heuristic algorithm is specifically defined to accommodate the newly AnyTraffic data group and to find the proper set of branch nodes of the minimum-cost network entity (e.g. tree).

This paper is organized as follows. Section II describes the three routing strategies considered on behalf of this study to forward an offered load of combined unicast and multicast traffic: the two traditional routing strategies and the outcome strategy introduced by this study. In Section III the formulation of the problem under study is done with some highlights on the needs for a novel heuristic algorithm for the proposed routing strategy. Section IV consists in a detailed mathematical formulation of the AnyTraffic routing algorithm, where a novel Steiner tree-based heuristic is specifically devised to accommodate the AnyTraffic data group. The algorithm pseudocode is also illustrated. Section V presents the performance analysis based on extensive experiments over a considerable set of network models. Conclusions are presented in Section VI.

II. Related Work and Contributions

In today's meshed networks relying on packet-switched technologies, multicast traffic can be carried by means of the following approaches:

The first approach (AP1) consists in setting up point-to-point data paths between each edge node and forward both multicast and unicast traffic over these switched paths. (Constraint-based) shortest path algorithm is commonly used to compute the corresponding path across the network topology. This implies that both unicast and multicast traffic is carried over point-to-point data paths (referred to as network trunks) between each edge-node pair. This approach also implies that the multicast traffic is replicated as many times as the number of edge nodes processing multicast traffic. This results in saving system resource for state maintenance at the expense of higher bandwidth consumption.

Although there is no need to entail dedicated resources in intermediate nodes to process the information that is not address to them, the usage of network bandwidth resources is clearly suboptimal. As can be observed in Fig.1 the two independent P2P data paths initiated at node s and with different destination nodes, namely $d1$ and $d2$, overlap in the first hop where, in the case of multicast traffic, the same information is transmitted two times over the same shared link.

The second approach (AP2) consists in setting up dedicated point-to-multipoint data paths for multicast traffic in addition to point-to-point data paths for unicast traffic. Steiner tree heuristics [5] are commonly used to construct the minimum cost tree dedicated to multicast traffic. Edge nodes in this case have to provide for differentiated treatment of incoming native multicast vs. unicast traffic such as to map it in the corresponding data path. Multicast traffic is mapped to P2MP data paths and unicast traffic is mapped into P2P data paths (see Fig.2). In the former case, the P2MP data paths can be either inclusive or selective. Inclusive implies that a single P2MP data path is setup for the entire set of multicast groups to a set of edge nodes. Note that the set of edge nodes may be greater than the number of edge nodes of each individual multicast group resulting thus in saving state at the expense of bandwidth waste. Selective P2MP (the case considered in this study) implies that each multicast group is mapped into a dedicated P2MP data path, resulting thus in saving bandwidth at the expense of system resource needed for additional P2MP state maintenance. This approach also implies that dedicated point-to-point data paths must be provisioned for unicast traffic.

Being the computation of a minimum-cost Steiner tree an NP-complete problem [6], in this study we employ the minimum cost path heuristic algorithm (MPH) [7] to compute a minimum-cost Steiner tree for a multicast connection. In MPH,

starting from a source node the tree is gradually grown until it spans all destination nodes belonging to a multicast group. The growth is usually based on the addition of shortest paths between destination nodes already in the tree and destination nodes not yet in the tree.

Traditionally, packet-switched technologies can carry multicast traffic using one of the above approaches. We propose the following novel approach (AP3). It can be seen as a refinement of the first approach i.e. single network entities are provisioned such as to carry both unicast and multicast traffic. However, in this case, point-to-point data path segments are appended to designated branch nodes toward edge nodes that belongs to a given set of one or more multicast groups. As shown in Fig. 3, both traffic are forward together over the same network entity till a branch node b1. From there, two appended P2P paths go toward the destination nodes of a AnyTraffic data group. Note that b1 in this case in different from the one of the second approach given by the Steiner tree algorithm.

Multicast traffic is marked at the ingress edge node using e.g. the MSB of the S-VLAN ID field in case of Ethernet. Other traffic techniques for discriminating between unicast and multicast traffic at branching nodes can be considered in case of other forwarding technologies. Based on this frame marking, branching nodes along the tree replicate the multicast traffic to the outgoing interfaces towards edge nodes registered for the corresponding multicast groups whereas unicast traffic is not replicated. The objective here is thus to benefit from the advantages of the traditional approaches while minimizing their respective drawbacks, i.e., keep the state maintenance overhead as low as possible while avoiding bandwidth waste by i) relying on replication of multicast traffic at branching points only (like in approach 2) and ii) keeping unicast traffic transmission over "as short as possible" data paths (like in approach 1).

To the best of our knowledge, there has not been any study suggesting a constraint-routing algorithm able to compute data paths allowing to forward unicast and multicast traffic together. Our contribution consists in i) designing and validating the algorithmic requirements to compute the path of the underlying data path structure (in the context of source-initiated routing) and ii) comparing the bandwidth and system resource consumption (for state maintenance) for a given set of multicast groups against the traditional approaches.

III. Problem Formulation

The issue to investigate is related with a novel heuristic algorithm that must be devised to accomodate the proposed routing approach.

Indeed, the direct application of one of the routing algorithms used by traditional approaches would result in an underperforming solution. For instance if we apply the Shortest Path algorithm, we would be underperforming in terms of multicast routing (as commented in the former Section II for the first approach). On the other hand, if we apply Steiner tree heuristic algorithm, we would be underperforming in terms of unicast routing because the path to carry the unicast traffic - one of the leaves of the multicast tree - would be too long when compared to the P2P shortest path, requiring considerable additional (unicast) bandwidth.

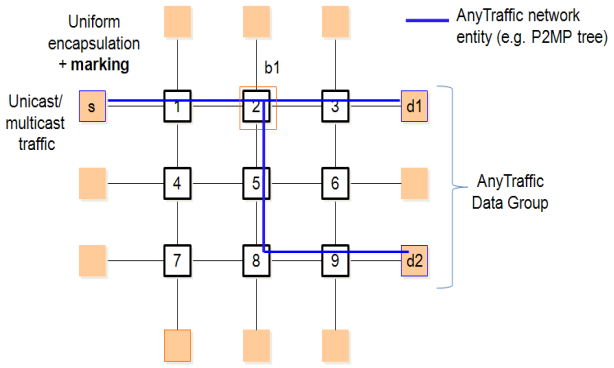

Fig. 3. The proposed AnyTraffic Approach: an AnyTraffic network entity (e.g. P2MP-tree) to carry both unicast and multicast traffic. Note that b1 might be different from b1 of the approach 2.

The concept of AnyTraffic data group is therefore introduced to define a group of destination nodes receiving unicast and multicast traffic over the same source-initiated network entity (Fig. 3). The novel heuristic algorithm, which is mathematical described in next section, attempts to construct a single network entity per each AnyTraffic data group. Hence, the aim of the heuristic algorithm is to find, at the minimum-cost, a set of branch nodes that takes into account unicast and multicast traffic constraints. At these designated branch nodes, several P2P data path segments are appended to reach the destination nodes that belong to a given set of one or more AnyTraffic data groups. The branch node selection is done according to a given pruning condition (e.g. see the approach followed in section IV) in order to guarantee a low increase of bandwidth consumption as well as of the length of unicast path. In fact, the created network entity is a root-initiated P2MP tree which is signaled using the technique described in [8]. Note that this single scheme for AnyTraffic data simplifies the management of the P2MP tree when considering dynamic multicast sessions.

In particular, our heuristic places itself between Shortest Path and Steiner tree algorithms, achieving better overall performance to forward an offered load consisting of combined unicast and multicast traffic in packet-switched meshed networks, as presented in Section V. In the next section, the mathematical description of the novel heuristic algorithm is presented.

IV. AnyTraffic Heuristic Algorithm

Consider a communication network modeled by a directed graph $G = (N, L)$, where N represents the set of nodes and L represents the set of links. Let $s \in N$ denote a source node. Each link $l \in L$ might have an associated capacity $b(l)$, distance $d(l)$, and cost $c(l)$. In Subsection A we present the method we use to calculate the cost $c(l)$ of link $l \in L$.

Let $\varphi_{s,M}$ denote a path request between source s and a group of destination nodes $M \subseteq N \backslash \{s\}, M \neq \varnothing$. If $|M| = 1$, $\varphi_{s,\{d\}} = \varphi_{s,M}$ is a path request with a single destination node d and $M = \{d\}$, i.e., a request of a P2P data path to unicast (one-to-one) traffic forwarding. Otherwise, it is a path request with multiple destination nodes $d_1, ..., d_{|M|}$, i.e., a request of a P2MP data path to multicast (one-to-many) traffic forwarding; hereafter, M is also refered to as an AnyTraffic data group.

Let $T_{s,M} = (N^*, L^*)$ be a connected subgraph without cycles of $G = (N, L)$, source-initiated by s, $M \subseteq N^* \subseteq N$, $L^* \subseteq L$. The objective of the AnyTraffic routing algorithm is to construct a graph $T_{s,M}$, for a given source s and AnyTraffic data group M, such that it supports both P2P data path $\varphi_{s,\{d\}}, d \in M$, and P2MP data path $\varphi_{s,M}$. The graph $T_{s,M}$ is built up by successive choices of a branch node $n^* \in N$ that meets a set of given conditions (explained below).

At a given source node, s, processing of a path request depends on the nature of the traffic, i.e, it is either a request to carry unicast traffic or a request to carry multicast traffic. Therefore, we have the following:

If a P2MP path request $\varphi_{s,M}$ arrives and there has already been created a graph $T_{s,M}$ (i.e., because another P2MP request $\varphi_{s,M}$ has been already served), the request is supported by $T_{s,M}$. Otherwise, the AnyTrafic routing algorithm is performed (described below) so that to establish a new tree. If a P2P path request $\varphi_{s,\{d\}}$ arrives, three different situations are possible to occur, namely (i) $d \in M$, $|M| > 1$, $T_{s,M}$ is already created and thus $\varphi_{s,\{d\}}$ is supported by $T_{s,M}$; (ii) $d \in M$, $|M| > 1$ but $T_{s,M}$ is not created yet and thus a shortest path must be setup; (iii) $d \notin M$ and thus a shortest path must be setup.

The AnyTraffic routing algorithm comprises two phases, namely an initialization phase and a tree computation phase. The initialization phase assigns initial values to algorithm attributes. The computation phase specifies the set of iterative processes that occur during the actual execution of the algorithm.

A. Algorithm Initialization Phase

The initialization phase of the algorithm consists in defining attributes and assigning initial values to them. Since these attributes depend only on the network topology, this phase is performed once (off-line) and their values can be stored in the memory of network elements.

Let $x_{i,j}$ denote the cost of the shortest path from node i to node j, $i \neq j$, computed using the Dijkstra algorithm and based on the link cost $c(l), l \in L$. The number of hops is used as tie breaker. Below we present the method we use to calculate the cost $c(l) \in Z_+$ of link $l \in L$. Firstly, we find a uniform segmentation of the maximum distance link $L_{\max}$ into $Q_{\max}$ intervals, where $Q_{\max}$ is the maximal node degree in the network. For instance, for the maximum link distance of $100km$ and the maximal node degree of 5 we have the following intervals: $]0, 20]$, $]20, 40]$, etc. Then, for each network link l we find the corresponding interval number i such that $\frac{(i-1)L_{\max}}{Q_{\max}} < d(l) \leq \frac{iL_{\max}}{Q_{\max}}$. Therefore, the link cost is calculated as:

$$c(l) = \left\lceil \frac{Q_{\max} + (i-1)}{Q(l)} \right\rceil, \tag{1}$$

where $Q(l)$ is the degree of the origin node of link l. Such assigned costs give preference to shorter links and their calculation involves a smoothing factor inversely proportional to the node degree. Note that this link cost calculation is possible only if the link distance metric, $d(l)$, is known.

Accordingly, $x_{s,d}, d \in M$, is the cost of the shortest path from source to destination. Among all path costs, we can find $d_{\max} = \max\{x_{i,j} : i, j \in N, i \neq j\}$, which corresponds to the shortest path of maximal cost in the network.

Let the function $\Psi(x) : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ be a function defined as follows:

$$\Psi(x) = x\left(1 + e^{-\frac{f(x)}{d_{\max}}}\right), \tag{2}$$

where function $f(x) : \mathbb{R}^+ \rightarrow \mathbb{R}$ is defined as

$$f(x) = \alpha x - \beta, \tag{3}$$

and $\alpha, \beta \in]0, 1]$.

The function $\Psi(x)$ is used to specify the threshold for the maximum cost of a path that is alternative to the path of cost x. In particular, $\Psi(x_{s,d})$ determines what is the acceptable cost deviation of an alternative path between nodes s and d from the path given by the Dijkstra algorithm, which is the best choice for unicast (one-to-one) traffic forwarding. This admissible cost deviation depends on a linear on a linear function $f(x)$ and d_{max}. Parameters α and β of $f(x)$ define the shape of the threshold function. The maximum growth is achieved at α approaching zero and β approaching one. In this study, after performing a number of experiments, we believe that $\alpha = 0.7$ and $\beta = 0.3$, are good values to initiate the algorithm so that the state consumption would be the lowest possible.

Function $\Psi(x)$ is applied (indirectly) as a decision criterion by the AnyTraffic routing algorithm in order to decide whether an alternative path between two nodes is acceptable or not. Closer to the tree-root/source, selection as part of the AnyTraffic subgraph of alternative paths (deviating from the shortest path) among all possible alternative paths is desirable up to a certain limit for unicast traffic. Indeed, for link cost metrics as defined in Eq.(1), increasing the path cost for such traffic results in additional state consumption that increases inversely to the "distance" from the tree-root: branching unicast traffic closer to the source consumes more states (than branching closer to one of the tree-leaves) at the expense of a slightly higher bandwidth consumption. The initial linear-growth for lower x is smoothly vanished into a curvature as x becomes larger, given the former idea. Having defined Ψ, we can calculate a Maximum Deficit factor $\Delta_{\max}^{s,d}$ for each initial (P2P) path $p_{s,d}, d \in M$, given by:

$$\Delta_{\max}^{s,d} = \Psi(x_{s,d}) - x_{s,d} = x_{s,d} e^{-\frac{f\left(x_{s,d}\right)}{d_{\max}}}. \tag{4}$$

This deficit factor represents what is the acceptable increment of the cost of an alternative path or, in other words, how

much the path can be deviated in order to process both unicast and multicast traffic without too much damage for the unicast traffic forwarding. Notice that since $x_{s,d}$ and $d_{\max}$ depend only on the topology, $\Delta^{s,d}_{\max}$ can be computated off-line, during the initialization phase of the algorithm, and its value is constant along the tree computation.

B. Algorithm Computation Phase

This phase of the algorithm consists in the tree computation itself, which is a progressive and iterative process. In order to make easier the understanding of the following description an illustrative example is depicted in Fig.4. A more detailed explanation of such example is presented in Subsection C.

Let's define leaf as the tuple $\omega_{v,\Lambda} = \{v, \Lambda\}$, where $v \in N$ is a leaf seed and $\Lambda \subseteq M$ is a subset of destination nodes. The initial leaf is the tuple $\omega_{s,M} = \{s, M\}$, where s is the seed root from where the computation is initiated and which comprises all destination nodes M. We define Ω as the set of leaves remaining to be processed; at the beginning, the set of leaves has only the initial leaf, $\Omega = \{\omega_{s,M}\}$. We also define the initial graph $T_{s,M} = (\{s\}, \varnothing)$.

For each $\omega_{v,\Lambda} \in \Omega$, the algorithm searches for a branch node $n^* \in N$, to be included in $T_{s,M}$, and such that source s is connected with a subset of nodes $\Lambda^* \in \Lambda$ through n^*. The algorithm terminates when there is no leaves left in Ω and all destinations $d \in M$ can be reached from s with the graph $T_{s,M}$. More specifically, at each iteration step an arbitrary leaf $\omega_{v,\Lambda}$ is pull out from Ω to be processed. For this leaf a set of candidate branch nodes A_ω is found. The set A_ω is restricted to unvisited nodes in previous iterations that are adjacent to v and have the node degree equal or greater than three, i.e., the nodes that have at least two outgoing links, apart from the outgoing link to node v. In case the node degree of an adjacent node n is two, the first node with node degree equal or greater than three and laying on a path going from v through n is included into A_ω. For instance, at the example depicted in Fig.4, $A_\omega = \{1, 2, 3\}$ for the scheme on the left.

At each candidate branch node $n \in A_\omega$ being evaluated (i.e. one of the adjacent nodes of v), one alternative paths per each $d \in \Lambda$, initiated at v but forced to pass through n, is computed, $p_{v,n,d}$. Each alternative path might introduce a deficit (additional cost), $\Delta^{v,n,d}_{local}$, when compared with previous path, $p_{v,d}$. A pruning condition is defined. This step of the algorithm is performed at each candidate branch node $n \in A_\omega$ of the currently processed leaf $\omega_{v,\Lambda}$ and for each $d \in \Lambda$, so that to determine if the set of alternative paths from v to each $d \in \Lambda$, and going through n, could be accepted.

First, for each $d \in \Lambda$, the cost $x_{n,d}$ of the shortest path from n to d is computed. Since the alternative path (forced to pass through n) might introduce additional cost when compared with cost $x_{v,d}$, such local deficit at node n is calculated:

$$\Delta^{v,n,d}_{local} = (x_{v,n} + x_{n,d}) - x_{v,d}. \tag{5}$$

There is a cumulative path deficit for each $d \in \Lambda$, that at node n sums up the local deficits produced by forcing the path

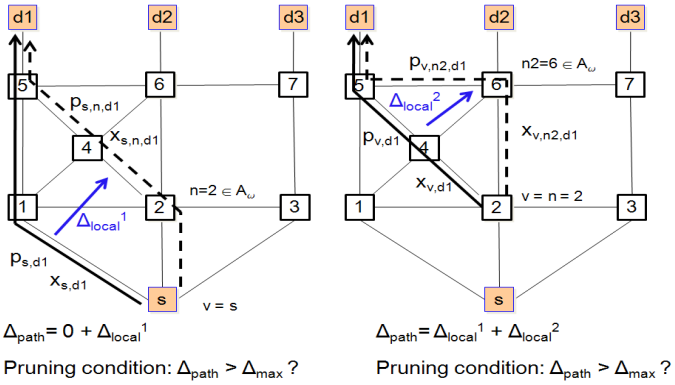

Fig. 4. Scheme on the left: initial leaf $\omega_{s,M}$, $M = \{d1, d2, d3\}$; first step of the algorithm. Scheme on the right: if previous pruning condition is satisfied and node 2 is selected as branch node, the process is repeat from the node 2 which is now the leaf seed of the leaf $\omega_{v,\Lambda}$, $\Lambda = \{d1, d2, d3\}$.

$p_{s,d}$ to go through already accepted branch nodes of $T_{s,M}$ and candidate branch node n:

$$\Delta^{s,d}_{path} = \sum_{i=0}^{u-1} \Delta^{i,j+1,d}_{local} = \sum_{i=0}^{u-1} (x_{i,i+1}) + x_{n,d} - x_{s,d}, \tag{6}$$

where $b_0 = s, b_1, ..., b_{n-1}, b_n = n$ are consecutive nodes on path $p_{s,n}$ such that $b_i \in T_{s,M}, i = 1...u-1$.

Then, and still for each $d \in \Lambda$, a comparison between the path deficit (6) and the Maximum Deficit Factor (4) is performed. If $\Delta^{s,d}_{path} > \Delta^{s,d}_{\max}$, the algorithm isolates node d so that to prune it from leaf $\omega_{v,\Lambda}$ and create a new leaf for further optimization. Otherwise, the alternative path can be accepted and joined to a common trunk that supports the nodes remaining in Λ.

When all candidate branch nodes have been evaluated, by running the pruning condition to each $d \in \Lambda$, the selection of the branch node, $n^* \in A_\omega$, can be done. The branch node decision is taken by considering the minimum total deficit among all candidate branch nodes $n \in A_\omega$. Each one of those nodes has its own deficit introduced by those $d \in \Lambda$ that meet the pruning condition. However, in order to have fairness it is not enough for branch node decision consider only the deficit based on the cost metric. Hence, we decided to ponderate the deficit of each candidate branch node at two levels. Firstly at the path level, by doing the sum of a fraction γ of the local deficit as defined in (5) and a fraction $(1 - \gamma)$ of local deficit also as defined in (5) but instead of taken the cost metric defined in (5), the calculus is done taken into account the number of hops (in order that more hops represents more state records at the nodes), $\Delta^{v,n,d*}_{local}$. Secondly at the node level, a fraction σ of the ratio P_j, consisting in the number of alternative paths where $d \in \Lambda$ meet the pruning condition, divided by the total number of paths that are possible to join (i.e. $|\Lambda|$), at $n \in A_\omega$, is subtracted from the former sum in order to favor those candidate branch nodes with more paths joined. Thus, for leaf $\omega_{v,\Lambda}$ we have:

$$\Delta^{n,\omega}_{candidate} = \sum[\gamma\Delta^{v,n,d}_{local} + (1-\gamma)\Delta^{v,n,d*}_{local}] - \sigma P_j \quad (7)$$

In this study, after performing a number of experiments, we select $\gamma = 0,5$ and $\sigma = 2$. However, this variables can be tuned in function of the bandwidth and state consumption objectives.

The candidate branch node n^* that has the lowest deficit, $n^* = \min\{\Delta^{n,\omega}_{candidate} : n \in A_\omega\}$, is selected as a branch node. Accordingly, $T_{s,M}$ is updated with all those links and nodes that lay on the path from v, which is the seed of the currently processed leaf $\omega_{v,\Lambda}$, to n^*. Note that the resulting graph is different from that given by the conventional Steiner tree algorithm. At this point two new leaves are created, namely, $\omega_1 = \{n^*, \Lambda_{n^*}\}$ and $\omega_2 = \{v, \Lambda \setminus \Lambda_{n^*}\}$. We add leaf ω_1 and ω_2 to the set Ω for further processing, respectively, if $|\Lambda_{n^*}| > 1$ and $|\Lambda \setminus \Lambda_{n^*}| > 1$. If either $|\Lambda_{n^*}| = 1$ or $|\Lambda \setminus \Lambda_{n^*}| = 1$, we update $T_{s,M}$ with all those links and nodes that lay on the shortest path, respectively, from n^* to $d \in \Lambda_{n^*}$ and from v to $d \in \Lambda \setminus \Lambda_{n^*}$. Note that the branch node n^* is excluded from the set of adjacencies of v, i.e., $A_{\omega_2} = A_\omega \setminus \{n^*\}$. As already mentioned, the branch selection cycle is repeated for each leaf left in Ω. The time complexity of this algorithm depends on the AnyTraffic group M and the set of candidate branch node Aw; the latter with an exponential dependency of the number of hops allowed by the maximum deficit factor of the each path joined to the previous selected branch node. This would be acceptable if the explored set $A_\omega << N$ (total number of network nodes).

Nonetheless, it is worth to mention that such complexity can be considerably reduced if the set Aw is restricted to those adjacent nodes that are within a given perimeter angle with respect to the node belonging to the shortest path. The pseudo-code of the AnyTraffic routing algorithm is given in Fig.5.

C. Example

In Fig.4, an illustrative example of two consecutive steps of the algorithm is depicted. The example shows the branch node evaluation mechanism to just one of the candidate branch nodes and to one destination node.

The scheme on the left represents the initial step. For instance, consider the initial leaf $\omega_{s,M}$ where s is the node processing the incoming path requests of both unicast and multicast traffic and the Anytraffic data group $M = \{d1, d2, d3\}$. The $node$ 2 is the current candidate branch node being evaluated from a set $A_\omega = \{1, 2, 3\}$. This set corresponds to the adjacent nodes of s with a node degree equal or higher than three. Thus, the computation of the local deficit, $\Delta^{s,n,d1}_{local}$, introduced by the alternative path, $p_{s,n,d1}$, is done. The cumulative path deficit associated to the original $p_{s,d1}$is update with the former local deficit and the pruning condition is verified. If path deficit is lower than the Maximum Deficit Factor for such path $p_{s,d1}$, $\Delta^{s,d1}_{path} > \Delta^{s,d1}_{\max}$, the alternative path can be accepted and joined to a common trunk that supports the nodes remaining in Λ.

```
INPUT: G, c(l), source s, M
OUTPUT: T_{s,M} = (N*, L*), N* ⊆ N, L* ⊆ L
INIT{
    ω_{s,M} = {s, M}
    Ω = {ω_{s,M}}
    p_{s,d} ⟵ SP(s, d)        ∀d ∈ M
    Δ_max(p_{s,d})            ∀d ∈ M
}
ANYTRAFFIC_HEURISTIC{
  while |Ω| ≠ ∅
    Δ_candidate = 0
    ω_{v,Λ} ⟵ Ω,        Ω = Ω \ {ω_{v,Λ}}
    A_ω = Adjacency nodes(v)
    for each n ∈ A_ω
        Λ_n ⟵ Pruning Condition(n, Λ)
        Δ^{n,ω}_candidate ⟵ Pdeficit (see Eq. (7))
    end
    n* = min{Δ^{n,ω}_candidate : n ∈ A_ω}
    if |Λ_{n*}| > 1 → ω_1 = {n*, Λ_{n*}}
    else ω_1 = ∅
    if |Λ \ Λ_{n*}| > 1 → ω_2 = {v, Λ \ Λ_{n*}}
    else ω_2 = ∅
    Ω = Ω ∪ {ω_1, ω_2}
    T_HEUR = T_HEUR ∪ p_{v,n*}
  endwhile
}
Function Pruning Condition(n, Λ){
    for each d ∈ Λ
        p_{n,d} ⟵ SP(n, d)
        Δ_path(d)_k = Δ_path(d)_{k-1} + Δ_local(d)_k
        if (Δ_path(d) > Δ_max) → Λ_n = Λ_n \ {d}
    endfor
return Λ_n
}
```

Fig. 5. The pseudo-code of the AnyTraffic heuristic algorithm being proposed.

Supposing that the candidate branch $node$ 2 is the branch node selected after all the other candidates were also evaluated, we have the situation depicted in the scheme on the right. At this point, the nodes s and $node$ 2 were added to $T_{s,M}$. Now, consider the leaf $\omega_{v,\Lambda}$ where v is $node$ 2 and is now the leaf seed. From this point, we repeat the same process described for the previous leaf. The candidate branch node considered here is the $node$ 6 from a set $A_\omega = \{1, 3, 4, 6\}$ and the alternative path $p_{v,n2,d}$. However, the path deficit is now the sum of both local deficits, $\Delta^{s,n,d1}_{local} + \Delta^{v,n2,d1}_{local}$. If it remains lower than the Maximum Deficit Factor, which is constant along the algorithm computation, the new alternative path can be accepted and joined. Otherwise, it is rejected and a new leaf is created and add into the set of leaves Ω - for further optimization.

V. Performance Analysis

The performance of the proposed heuristic algorithm is evaluated and compared against the two strategies described in Section II in terms of bandwidth consumption and system resource consumption.

In order to assess the robustness of the proposed solutions and determine the corresponding dependencies, several network topologies were simulated. Because similiar behaviour was observed, we only present results for the following

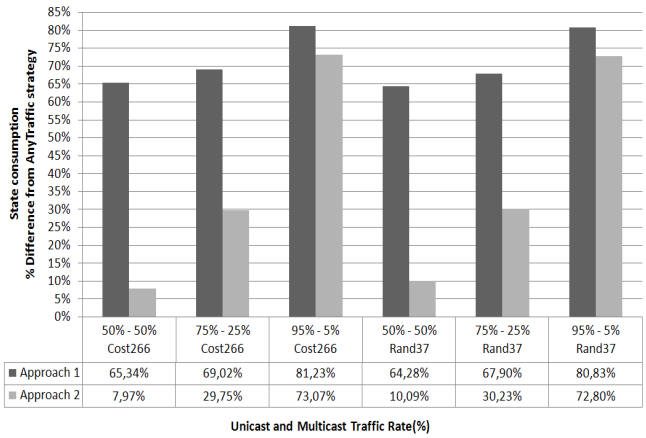

Fig. 6. State Consumption for the networks Cost266 and Rand37 of 37 nodes.

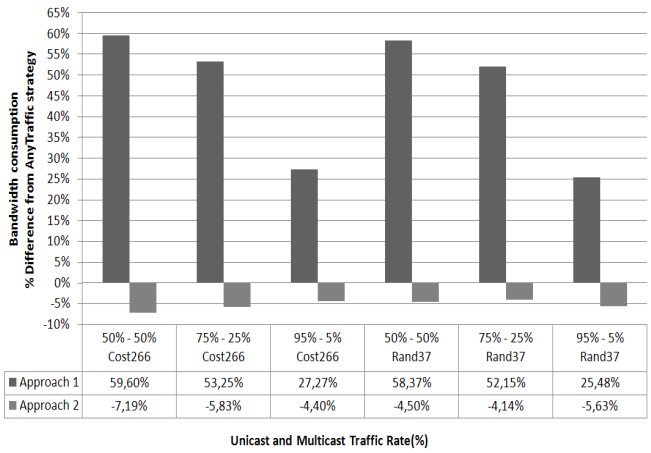

Fig. 7. Bandwidth Consumption for the networks Cost266 and Rand37 of 37 nodes.

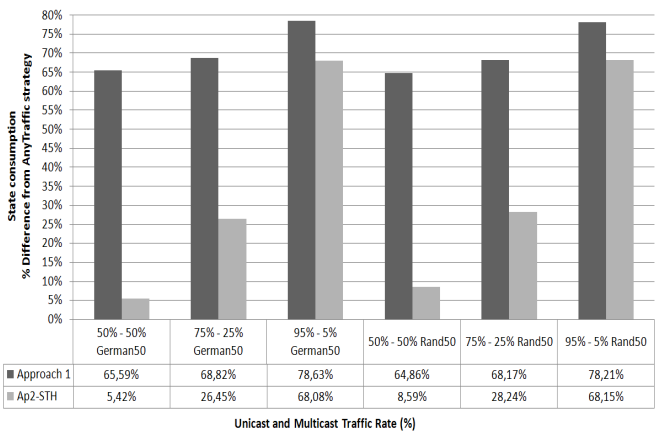

Fig. 8. State Consumption for the networks German50 and Rand50of 50 nodes.

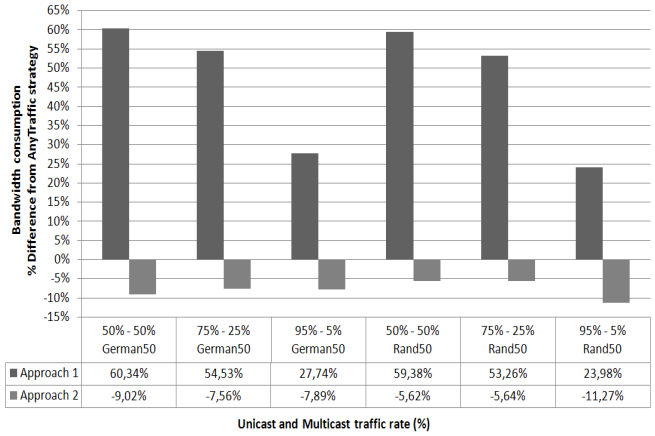

Fig. 9. Bandwidth Consumption for the networks German50 and Rand50 of 50 nodes.

networks: Cost266 [9] and Rand37 with 37 nodes, and German50 [9] and Rand50 with 50 nodes. The differenciating properties of these topologies consist in different node degrees and clustering coefficient (c.c.) ranging in the interval [0,1], besides the number of nodes and links. Regarding the 37-nodes network topologies, the c.c. is 0.0 to the Cost266 and 0.2489 to the Rand37. Regarding the 50-nodes networks topologies, the c.c. is 0.19 to the German50 and 0.2752 to the Rand50. The Rand networks are generated based on the generator algorithm proposed in [10], where the graphs are randomly created from a sequence of node degrees. The traffic generation is a bound and discrete process. The unicast and multicast traffic are both generated within a bound range of two discrete traffic classes, namely class 1 - MPEG-4 standard definition (SD) - of 2 Mbps and class 2 - MPEG-4 high definition (HD) - of 8 Mbps.

Each node in the network is an ingress-egress node generating 150 connection requests. Different percentages of unicast and multicast traffic ration are considered, namely 50%-50%, 75%-25% and 95%-5%. For each multicast connections/sessions, the size of the destination nodes ranges between a minimum of $log_2(N)$ and a maximum of $[log_2(N)]^2$, where N represents the set of nodes in the network. The simulations are performed in a non-blocking regime where enough network resource availability is assumed. In order to settle the bounds of the algorithm, we have simulated a best case where all multicast requests are processed first, creating the network entities (e.g. P2MP trees) for the AnyTraffic data groups and then process the unicast requests looking for the minimum cost path among all those trees.

A. Results

We have defined Relative Gain as the percentage in network performance gain (in terms of either bandwidth or state consumption metrics) achieved when the AnyTraffic routing scheme is used versus one of the traditional routing schemes here taken as references namely, approach 1 and approach 2 (see Section II for details). The Relative Gain is defined as follows:

$$\text{Relative Gain } [\%] = \frac{APx - AP3}{APx} * 100 \qquad (8)$$

where the index $x = 1$ refers to the approach 1 using SP algorithm to both traffic forwarding and the index $x = 2$ refers to the approach 2 using SP algorithm to unicast traffic forwarding and Steiner Tree heuristic MPH to multicast traffic forwarding. Therefore, the simulation results are displayed as the difference in percentage from AP1 and AP2 with respect to the Anytraffic approach, in function of the unicast and multicast traffic rate generated in the network.

Figures 6 and 7 show the results obtained for the networks of 37 nodes, namely Cost266 and Rand37, in terms of state and bandwidth consumption, respectively. As it can be seen from those figures, the proposed approach (AP3) has an outstanding performance in terms of state consumption compared with both AP1 and AP2 to the range of 50%-95% of unicast traffic rate. As unicast traffic rate increases, the gain is higher. From the figures above, for the interval of 75%-25% to 95%-5% of traffic rate in both pair of networks (37 and 50 nodes), we have state consumption gains ranging from around 30% to 70%. Even though the network entities created by the multicast traffic requests are fewer, there is more unicast traffic that goes over them (i.e. network entities), reducing the number of states consumed with respect to AP1 and AP2. In terms of bandwidth consumption, the AP3 has worst performance due to the longer paths that unicast traffic has to follow. The tendency of bandwidth consumption is inversely to the state consumption. The additional bandwidth decreases because with less anytraffic entities created by multicast requests, the unicast traffic goes more often by P2P (shorstest) data paths, becoming closer to the AP1 and AP2 values. However, this does not invalidate that a considerable amount of unicast traffic is still carried by means of AnyTraffic trees as said before. Nevertheless, these values can be improved at the expense of decreasing a fraction of the state consumption gain by tunning the algorithm (e.g. changing the values of γ and σ).

The same behaviour is observed for the networks of 50 nodes, namely German50 and Rand50, shown in Figures 8 and 9. However, results are a bit less favorable compared to the results obtained with Cost266 and Rand37. This reflects that more nodes with higher node degree influence the performance of the algorithm. Although, the bandwidth consumption here is a little higher, regarding the considerable gains obtained for state consumption, we can decrease the bandwidth consumption again by tuning the algorithm to lower the aggregation of data paths. For instance, for the German50 network with 75%-25% of traffic rate, the AP3 has around 27% of state consumption gain when compared with AP2 versus an additional bandwidth consumption of around 8%. It is worth to note that in both pair of networks we can see that the networks with higher clustering coeffcient (those Rand networks), the algorithm improves with respect to the network with the same number of nodes but lower clustering coefficient.

VI. Conclusion

In this study we have outcome with a novel routing approach aiming to keep the system resource consumption (for state maintenance) overhead as low as possible while avoiding bandwidth waste by i) relying on replication of multicast traffic at branch points only (like in approach 2) and ii) keeping unicast traffic transmission over "as short as possible" data paths (like in approach 1). The concept of AnyTraffic data group was introduced and a specific heuristic algorithm was devised to accomodate this new routing approach.

The AnyTraffic routing algorithm was then compared against traditional schemes routing using extensive simulation experiments to demonstrate the effectiveness of the proposal. The simulation results obtained are very satisfatory, settling the algorithm bounds: it gives a minimum improvement bound on state consumption and higher bound on bandwidth utilization increase. Therefore, further work is expected addressing issues like (1) dynamic multicast sessions where a edge node receiving traffic can be joined and released of a multicast group, (2) adapting the routing algorithm to optimize those cases of P2MP-tree (partial) overlapping where the same source-initiated multicast groups share almost the same nodes. This will improve the state consumption results; and (3) study a blocking regime scenario and refine the proposed algorithm in order to achieve load-balancing and dynamical use of available bandwidth resources.

Acknowledgment

The authors would like to thank the grant SFRH / BD / 36950 / 2007 provided by the Portuguese Government Entity, Fundação para a Ciência e a Tecnologia (FCT). The work described in this paper was carried out with the support of the BONE-project ("Building the Future Optical Network in Europe"), a Network of Excellence funded by the European Commission through the 7th ICT-Framework Programme, and with the support of the COST 293 Action.

References

[1] Rosen, E., Ed., Multiprotocol Label Switching (MPLS) Architecture, RFC 3031, January 2001

[2] Manie, E., Ed., Generalized Multi-Protocol Label Switching (GMPLS) Architecture, RFC 3945, October 2004

[3] Rouskas, G., Scheduling of combined unicast and multicast traffic in broadcast WDM networks,In Proceedings of Performance of Information and Communication Systems (PICS), 1998

[4] Singhal, N., Ed., Optimal Multicasting of Multiple Light-Trees of Different Bandwidth Granularities in a WDM Mesh Network With Sparse Splitting Capabilities, IEEE/ACM Transactions on Networking, Vol.14, NO.5, October 2006

[5] Hwang, F. K., Ed., The Steiner Tree Problem, Annals of Discrete Mathematics, Volume 53, Amsterdam, North-Holland Editions, 1992

[6] Garey, M. R., Graham, R. L., and Johnson, D. S., The complexity of computing Steiner minimal trees, SIAM J. Appl. Math., vol. 32, no. 4, pp. 835–859, 1977

[7] Takahashi, H. and Matsuyama, A., An approximate solution for the Steiner problem in graphs, Math. Japonica, pp. 573–577, 1980

[8] Aggarwal, R., Ed., Extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE Label Switched Paths (LSPs), RFC 4875, May 2007

[9] Orlowski, S., Pi'oro, M., and Tomaszewski, A. and Wessly, R., Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, April 2007, Belgium, url {http://sndlib.zib.de}

[10] Kim, H., Ed., On realizing all simple graphs with a given degree sequence, Discrete Mathematics, April 2008